HW: LATER model exercise

***Disclaimer: this code was generated exclusively using chatGPT and doesn't work because I have no idea what I'm doing and got stuck

```
% Load data
raw_data = load("WZ_RT_accuracy.mat");

% Preprocess data
% Code below is taken from Josh's github
    % Define selection criteria ("L" for "logical array"):
    %  1. Correct trials only (basic LATER model doesn't account for errors)
    %  2. Remove outlier RTs (need to check with Tim Kim about the
conditions
    %        that gave rise to super-long RTs)
    Ltrials = percorrSum == 1 & tRxnSum > expressCutoff(1) & tRxnSum < 1.2;
```

```
Unrecognized function or variable 'percorrSum'.
```

```
    % Now loop through and get 4 data sets (see Fig. 2
    %  in Kim et al):
    %  C_L,0:  Left choices, change-point trials
    %  C_L,1+: Left choices, non-change-point trials
    %  C_R,0:  Right choices, change-point trials
    %  C_R,1+: Right choices, non-change-point trials
    data_ = { ...
        tRxnSum(Ltrials & numdirSum == -1 & labelSum == 1), ...
        tRxnSum(Ltrials & numdirSum == -1 & labelSum ~= 1), ...
        tRxnSum(Ltrials & numdirSum ==  1 & labelSum == 1), ...
        tRxnSum(Ltrials & numdirSum ==  1 & labelSum ~= 1)};

    if nargout > 1
        labels_ = {'Left Choice, No CP', 'Left Choice, CP', 'Right Choice,
No CP', 'Right Choice, CP'};
    end
```

```
% Compute the reciprocal of RTs
reciprocalRTs = 1 ./ correctTrials(:, 1);

% Optionally, group RTs by condition or other factors
% (Assuming the condition is in column 2)
conditions = unique(correctTrials(:, 2));
groupedRTs = cell(length(conditions), 1);
for i = 1:length(conditions)
    groupedRTs{i} = reciprocalRTs(correctTrials(:, 2) == conditions(i));
end
```

```
% Now can use the groupedRTs for model fitting
```

```
% Define the LATER objective function
laterErrFcn = @(fits) computeNegativeLogLikelihood(fits, RTs);

% Define initial conditions and bounds
lowerBounds = [0.001 0.001];
upperBounds = [1000 1000];
initialValues = [initial_muR, initial_deltaS]; % Set initial values

% Create optimization options
opts = optimoptions(@fmincon, ...
    'Algorithm', 'active-set', ...
    'MaxIter', 3000, ...
    'MaxFunEvals', 3000);

% Define the optimization problem
problem = createOptimProblem('fmincon', ...
    'objective', laterErrFcn, ...
    'x0', initialValues, ...
    'lb', lowerBounds, ...
    'ub', upperBounds, ...
    'options', opts);

% Create a GlobalSearch object
gs = GlobalSearch;

% Run the optimization
[fits, nllk] = run(gs, problem);
```